Prototype Generation for Multi-label Nearest Neighbours Classification

Stefanos Ougiaroglou[1], Panagiotis Filippakis[2], and Georgios Evangelidis[3]

[1] Department of Digital Systems, School of Economics and Technology, University of the Peloponnese, 23100, Sparta, Greece
stoug@uop.gr
[2] Department of Information and Electronic Engineering, School of Engineering, International Hellenic University, 57400 Sindos, Thessaloniki, Greece
filipp1977@gmail.com
[3] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 54636 Thessaloniki, Greece
gevan@uom.gr

Abstract. Numerous Prototype Selection and Generation algorithms for instance based classifiers and single label classification problems have been proposed in the past and are available in the literature. They build a small set of prototypes that represents as best as possible the initial training data. This set is called the condensing set and has the benefit of low computational cost while preserving accuracy. However, the proposed Prototype Selection and Generation algorithms are not applicable to multi-label problems where an instance may belong to more than one classes. The popular Binary Relevance transformation method is also inadequate to be combined with a Prototype Selection or Generation algorithm because of the multiple binary condensing sets it builds. Reduction through Homogeneous Clustering (RHC) is a simple, fast, parameter-free single label Prototype Generation algorithm that is based on k-means clustering. This paper proposes a RHC variation for multi-label training datasets. The proposed method, called Multi-label RHC (MRHC), inherits all the aforementioned desirable properties of RHC and generates multi-label prototypes. The experimental study based on nine multi-label datasets shows that MRHC achieves high reduction rates without negatively affecting accuracy.

Keywords: Multi-label classification, Data Reduction, Prototype Generation, k-NN Classification, Binary Relevance, RHC, BRkNN

1 Introduction

Multi-label classification [13] is a challenging problem that has attracted the interest of the machine learning and data mining research communities. Contrary to traditional single-label classification problems, where an instance may belong to only one class, in multi-label problems an instance may belong to

more than one classes (or labels). Nowadays, multi-label classification is needed in numerous real-life applications of different problem domains. Typical examples of multi-label classification problems are image, text, protein, music and video/movies classification. For instance, a movie can simultaneously be "Crime" and "Drama". A music track may belong to more than one genre. An image may depict "mountain", "sea" and "beach".

The k-Nearest Neighbours (k-NN) classifier [5] is a typical instance-based (or lazy) classification algorithm. For each unclassified instance x, it searches the available training data and retrieves the k nearest to x instances. These instances are called nearest neighbours. Then, x is classified by a majority vote. In effect, x is classified to the most common class among the classes of the retrieved k nearest neighbours. The k-NN classifier is simple, easy to implement and has good performance. Moreover, it can be easily adapted for multi-label datasets. However, the k-NN classifier is CPU and memory intensive since all distances between a new unclassified instance and all training instances must be computed, and all training instances need to always be available in memory. Therefore, it cannot deal with large volumes of training data. Thus, in single-label classification, k-NN is usually applied in conjunction with a Prototype Selection (PS) [7] or Prototype Generation (PG) [12] algorithm.

PS and PG algorithms replace the initial training dataset by a small representative set. This set is called the condensing set and is used by k-NN to achieve comparable accuracy as when using the initial training dataset but at a much lower computational cost. PS algorithms select instances (or prototypes) that represent as much as possible the initial training set. The algorithms of this category are also called condensing algorithms. In contrast, PG algorithms generate prototypes by summarizing similar training instances of the same class. The prototypes generated by a PG algorithm are artificial instances that represent a specific data area in the metric space. Most of PS condensing and PG algorithms are based on the following simple idea: only the close to the class decision boundaries training instances are needed for classification tasks. The training instances that lie to the "internal" area of a class (far from decision boundaries) burden computational cost of the classifier, do not contribute to accurate classification and can be safely removed without loss of accuracy. Therefore, PS condensing and PG algorithms try to select or to generate a sufficient number of prototypes that lie close to class decision boundary data areas.

A sub-category of PS algorithms are called editing algorithms. Editing aims to improve accuracy rather than reduce data. To achieve this, editing algorithms try to improve the quality of the training data by removing noise and by smoothing the decision boundaries between the distinct classes. Of course, in case of non-separable classes, editing may fail to achieve these goals. Thus, contrary to PS condensing algorithms that condense the data by keeping only the instances that lie close to class decision boundaries, PS editing algorithms "clear" the class decision boundary areas by removing the close to the decision boundary instances. Although editing has a completely different goal, it can be used to improve the performance of PS condensing and PG algorithms. More specifi-

cally, the size of the condensing set built by PS condensing and PG algorithms depends on the level of noise in the training data. High levels of noise in the training set prevent many PS condensing or PG algorithms from achieving high reduction rates on the training data. In other words, the condensing set does not become small enough with respect to the original training set. Therefore, effective application of such algorithms requires removal of noise from the data, i.e., application of an editing algorithm beforehand.

PS and PG algorithms are appropriate for single-label classification problems. The Label Powerset (LP) transformation technique [13] could be an intuitive approach in order to use a PS or PG algorithm in a multi-label problem. LP transforms a multi-label dataset into singe-label datasets by considering each label combination (labelset) as a distinct class. However, LP can be applied only if the number of labels and the possible labelsets are small and there is a sufficient number of instances for each labelset. Otherwise, the total number of different combinations may increase exponentially, the reduction rate will be low and some combinations may be poorly represented.

Binary Relevance (BR) is the most widely-used problem transformation technique for multi-label problems. It transforms the multi-label problem into multiple binary problems, one for each label. A binary problem is a single-label problem with two labels. More specifically, for each label $l \in L$, BR builds a classifier in order to predict whether an instance belongs to l or not. In effect, BR copies the original training set $|L|$ times. Each copy concerns a different label l and contains all instances of the original training set, labelled as "1" if the original instance is labeled by l and as "0" otherwise.

The k-NN classifier in conjunction with BR is called BRkNN [6] and seems to be an ideal combination because k-NN classifier is a lazy classifier and does not build any classification model. BRkNN does not make $|L|$ copies of the training set. When an instance x needs to be classified, BRkNN searches for the k nearest to x neighbours once, like the single-label k-NN does. Then, the nearest neighbours voting procedure is repeated $|L|$ times, once for each label, and BRkNN makes $|L|$ label predictions for x. Therefore, x obtains the predicted labels of each voting procedure. Since each voting procedure concerns a binary classification problem, k should be odd to prevent ties.

The k-NN classifier stops being lazy when it is used in conjunction with a PS or PG algorithm. In effect, the condensing set is a classification model. In multi-label classification, if BR is used, one condensing set must be constructed for each label. Therefore, the goal of data reduction is not achieved and the k-NN classifier must search for nearest neighbours in each condensing set in order to make each individual label prediction. Hence, the computational cost remains high. This observation makes clear that the PS and PG algorithms must be adapted so that they can be used for multi-label datasets and this is the motive behind the present work. This paper proposes a PG algorithm for multi-label datasets. It constitutes a variation of Reduction through Homogeneous Clustering (RHC) [10], which is a fast and parameter-free PG algorithm for single label classification problems. The proposed algorithm is called Multilabel RHC

(MRHC). It inherits the aforementioned desirable properties of RHC and builds a multi-label condensing set that is then used by BRkNN. The experimental study conducted shows that MRHC achieves noteworthy reduction rates while classification accuracy is not affected.

The rest of this paper is organized as follows: Section 2 briefly presents the related work, while Section 3 reviews RHC. Section 4 presents the proposed MRHC algorithm. Section 5 presents the experimental study, and, Section 6 concludes the paper and gives directions for future work.

2 Related work

While most of research works in multi-label classification focus on proposing accurate classification algorithms, the issue of the computational cost on large multi-label training sets is somehow marginalized. There are only few research works that focus on speeding-up lazy classifiers in large multi-label training sets and even fewer that concern condensing algorithms for that type of dataset. As far as we know, there are no PG algorithms in the literature for multi-label training sets. In this section, we review the limited related works for fast multi-label classification.

The algorithm presented in in [4] can be considered as the first PS algorithm for multi-label datasets. However, it focuses on editing of imbalanced datasets. In effect, the authors presented an under-sampling method for imbalanced training sets inspired by the Edited Nearest Neighbour rule (ENN-rule) [15]. Kanj et al. in [8] proposed a PS editing algorithm also based on ENN-rule. The proposed algorithm uses Hamming loss to determine the noisy instances. The idea behind the algorithm is simple: the instances with high Hamming loss probably lie in close decision boundaries and for this reason they should be removed, like in the case of ENN-rule. Both aforementioned works concern data editing. Thus, they are not considered further in the present work.

To the best of our knowledge, the work presented in [1] is a first attempt that adapts existing PS algorithms that edit and condense multi-label data. In this work, the previous proposed PS LSBo and LSSm algorithms [9] are adapted. LSBo condenses the data while LSSm edits it. The proposed algorithms as well as their predecessors are based on the concept of local sets [3] and on the LP transformation technique. In single-label problems, the local set of an instance x is the largest set of instances centered on x, so that all instances are of the same class. In multi-label datasets, the authors define that there is no need for a local set to contain the exact same labelset. The labelset of the instances in a local set may slightly differ. The authors use the Hamming loss calculated over labelsets to measure the difference in the labelsets. If the Hamming loss between the labelsets of two instances is greater than a pre-specifed threshold, the instances are considered to be of different "classes". The proposed PS algorithms are not parameter-free and their performance depends on the pre-specified threshold. Therefore, they are not considered further in this paper.

The paper in [2] uses BR, LP and other transformation methods in conjunction with single label PS algorithms. In case of BR and its variants, the proposed strategy copies the original training set $|L|$ times. Each copy concerns a different label l and contains all instances of the original training set, labelled as "1" if the original instance labeled by l and as "0" otherwise. Then, the strategy utilizes a PS algorithm on each copy and builds $|L|$ condensing sets for each label. Every time an instance is selected, it receives a vote that is accumulated in a vector with the votes for all instances. The strategy builds a complete condensing set by selecting all instances with a number of votes that exceeds a pre-specified threshold. As we mentioned before, we are interested in parameter-free approaches. Hence, these strategies are not considered further in the present work. Furthermore, the same paper [2] points out the LP drawbacks, mentioned in Section 1, when it is used in conjunction with a PS algorithm.

The work presented in [11] proposes a scalable lazy classifier for large multi-label datasets. The authors proposed an implementation that takes advantage of the GPU architecture. In effect, the work implements MLkNN [16] on GPUs. The proposed method implements the execution stages of MLkNN in parallel on GPUs, offering computational speedup without loss of accuracy.

3 The Reduction through Homogeneous Clustering (RHC) algorithm

Reduction through Homogeneous Clustering (RHC) [10] is a PG algorithm that is based on the well known k-means clustering algorithm. RHC is parameter-free, hence the size of the condensing set is determined automatically. Also, it is a fast algorithm since it avoids costly and time-consuming pre-processing tasks on the training set, which may be prohibitive for large datasets.

Initially, RHC computes the class representatives by averaging the instances of each class. If a dataset has ten classes, RHC will compute ten class representatives. These class representatives are used as initial means for k-means clustering. Then, k-means discovers as many clusters as the number of classes in the training set. If a discovered cluster has instances of only one class (i.e., it is homogeneous), the cluster centroid is placed in the condensing set as a prototype. Otherwise (i.e., the cluster is non-homogeneous), the aforementioned procedure is recursively repeated on that cluster. RHC terminates when all the discovered clusters are homogeneous.

The centroid/representaive m of each cluster/class C is computed by averaging the n attribute values of instances x_i, $i = 1, 2 \ldots |C|$ that belong to C. More formally, the n attributes $m.d_j$ of m is estimated as follows:

$$m.d_j = \frac{1}{|C|} \sum_{x_i \in C} x_i.d_j, j = 1, 2, \ldots, n$$

Obviously, RHC generates more prototypes for the close to the decision boundary areas and fewer prototypes for the "internal" class areas. By using

the class representatives as initial means for k-means clustering, RHC increases the probability of quickly finding large homogeneous clusters and achieving a high reduction rate without costly k-means iterations (the larger the homogeneous clusters constructed, the higher the reduction rate achieved). RHC can become even faster if k-means clustering without full cluster consolidation is used. Full clusters consolidation means that k means clustering stops only when there are no moves of instances among clusters.

Contrary to many PS and PG algorithms, RHC always builds the same condensing set regardless the order of the data in the training set. Moreover, RHC is simple and quite easy to implement. The experimental study presented in [10] shows that RHC is faster and achieves higher reduction rates and than state-of-the-art PS condensing and PG algorithms without harming classification.

4 The Proposed Multi-label RHC (MRHC) Algorithm

The Multi-label Reduction through Homogeneous Clustering (MRHC) algorithm is a variant of RHC for multi-label training data. It works quite similar to RHC. However, MRHC builds a multi-label condensing set. The key question that should be answered is how to define homogeneity on clusters that contain multi-label data. In the case of MRHC, a cluster is considered homogeneous, when it contains instances that share at least one common label.

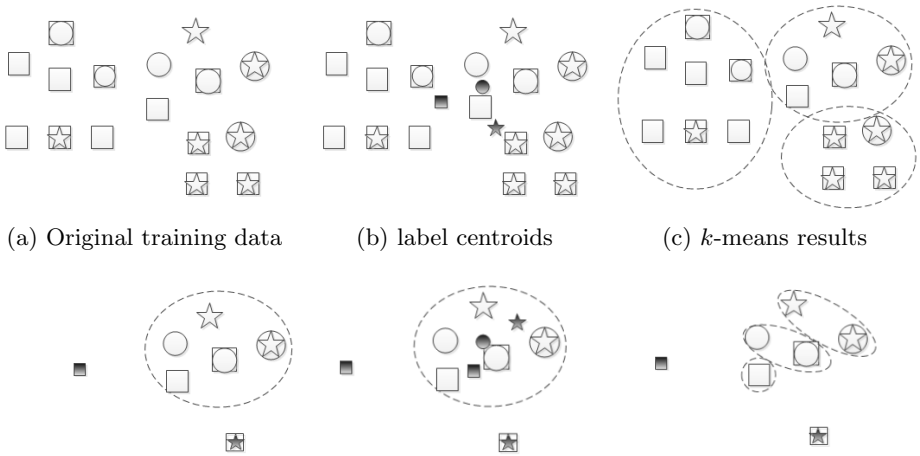
This is how the MRHC algorithm works. Initially, it builds a mean (representative) for each label l by averaging the instances that their label-set contains l. Then, it runs k-means clustering and discovers as many clusters as the number of labels in the training set. For each cluster C with instances that do not share a common label (i.e., C is non-homegenous), the aforementioned procedure is repeated considering only the instances that belong to C. For each homogeneous cluster, MRHC stores the cluster centroid in the condensing set as a prototype. The generated prototype is labeled by the common label(s) in C along with each label that appears in the label-set of more than half the instances in C. We call such labels majority labels. Like the case of RHC, MRHC terminates when all clusters become homogeneous.

Suppose a training set has three attributes (a,b,c) and three labels (x,y,z). Moreover, suppose that MRHC discovered a cluster C which contains the instances $inst_1$, $inst_2$ and $inst_3$ with the following BR representation:

- $inst_1$: $\{a, b, c, x, y, z\} = \{2, 1, 1, 0, 1, 1\}$
- $inst_2$: $\{a, b, c, x, y, z\} = \{1, 2, 3, 0, 1, 0\}$
- $inst_3$: $\{a, b, c, x, y, z\} = \{3, 2, 1, 1, 1, 1\}$

Since all instances contain label y, C is homogeneous. The label-set of the generated prototype p will be $\{y, z\}$ because y is the common label that renders C homogeneous and z is a majority label in C. Consequently, in a BR representation, p is $\{a, b, c, x, y, z\} = \{2, 1.67, 1.33, 0, 1, 1\}$.

Figure 4 represents a two dimensional example. Suppose that a dataset contains sixteen instances (Figure 4(a)). MRHC computes a representative for the



(a) Original training data (b) label centroids (c) k-means results

(d) Prototypes for the homogeneous clusters (e) label centroids for non-homogeneous cluster (f) k-means results

(g) final condensing set

Fig. 1. MRHC excution example

squares, a representative for the circles, and, a representative for the stars (Figure 4(b)). Then, k-means clustering uses the three label representatives as initial means and discovers three clusters (Figure 4(c)). Two of them are homogeneous because their instances have a common class. Thus, the cluster centroids constitute prototypes (Figure 4(d)). One prototype is labeled only by "square" because there is no majority label in the cluster. The other prototype is labeled by "square" and "star", because "star" is the common label and "square" is a majority label in a that cluster. For the instances of the non homogeneous cluster, MRHC recursively builds three homogeneous clusters (Figures 4(e,f)). Consequently, three more prototypes are stored in the condensing set. Thus, the final condensing set contains five prototypes instead of the sixteen instances of the initial training set (Figure 4(g)).

Algorithm 1 shows a non-recursive MRHC implementation. It uses a queue data structure, Q, to hold clusters. Initially, the whole training set is an unpro-

Algorithm 1 MRHC

Input: TS
Output: CS

```
 1: Q ← ∅
 2: Enqueue(Q, TS)
 3: CS ← ∅
 4: repeat
 5:    C ← Dequeue(Q)
 6:    if all the instances in C have at least one common label then
 7:       r ← centroid of C
 8:       r_labelset ← ∅
 9:       for each label l in C do
10:          n ← count the instances ∈ C with l in their labelset
11:          if  n > |C/2|  then
12:             r_labelset ← r_labelset ∪ l
13:          end for
14:       end for
15:       CS ← CS ∪ {r}
16:    else
17:       M ← ∅ {M is the set of label-centroids}
18:       for each label l ∈ C do
19:          m_l ← centroid of l
20:          M ← M ∪ {m_l}
21:       end for
22:       NewClusters ← K-MEANS(C, M)
23:       for each cluster C ∈ NewClusters do
24:          Enqueue(Q, C)
25:       end for
26:    end if
27: until IsEmpty(Q)
28: return  CS =0
```

cessed cluster and is placed in q (line 2). At each repeat-until iteration, MRHC dequeues cluster C from Q (line 5) and checks whether C is homogeneous (has at least one common label) or not. If it is (line 6), its centroid is placed in the condensing set (CS) as a prototype (line 15) labeled by the common and majority labels in C (lines 9-14). Otherwise, MRHC computes a list of label-representatives (M), one for each of the labels that exist in C (lines 1721). Then, MRHC calls k-means clustering, with parameters the non-homogeneous cluster C and the list of the initial label-representatives M to be used as initial means. The result is a new set of unprocessed clusters ($NewClusters$) (line 22) all of which are put into Q (lines 2325). The repeat-until loop continues until Q becomes empty (line 27), i.e., there are no more clusters to process.

MRHC inherits all the properties of RHC. Therefore, it is a fast and parameter-free PG algorithm. Also, MRHC builds the same condensing set regardless the order of the instances in the training set. MRHC can be ideally combined with BRkNN. For each unclassified instance x, the BRkNN classifier runs over the

condensing set (instead of the initial large training set) once and retrieves the k nearest to x prototypes. Then, the label-set of x is predicted by as many voting procedures as the number of labels. For each label l, the same k retrieved nearest prototypes vote in order to predict if x is labeled by l or not.

5 Performance Evaluation

5.1 Datasets

The performance of MRHC was evaluated by conducted experiments on nine multilabel datasets distributed by Mulan datasets repository [14][4]. Table 1 summarizes the key characteristics of the datasets used. The last two columns present the cardinality and the density of the datasets. Cardinality is the mean of the number of labels of the instances. Density is the mean of the number of labels of the instances divided by the number of labels. The second column of Table 1 lists the domain of each dataset.

Table 1. Dataset characteristics

Datasets	Domain	Size	Attributes	Labels	Cardinality	Density
CAL500 (CAL)	Music	502	68	174	26.044	0.150
Emotions (EMT)	Music	593	72	6	1.869	0.311
Water quality (WQ)	Chemistry	1060	16	14	5.073	0.362
Scene (SC)	Image	2407	294	6	1.074	0.179
Yeast (YS)	Biology	2417	103	14	4.237	0.303
Birds (BRD)	Sounds	645	260	19	1.014	0.053
CHD49 (CHD)	Medicine	555	49	6	2.580	0.430
Image (IMG)	Image	2000	294	5	1.236	0.247
Mediamill (MDM)	Video	43907	120	101	4.376	0.043

5.2 Experimental Setup

MRHC was coded in C and it runs only as a pre-processing step to build the condensing set. BRkNN was implemented in Python. We compared the performance of BRkNN running over the condensing set built by MRHC against the performance of BRkNN running over the original training set. The Euclidean distance was used as the distance metric. We measured two metrics: (i) Hamming loss and (ii) Reduction Rate, that were derived via a five-fold-cross-validation schema. Initially, we normalized the datasets in the $[0-1]$ range, and then, we split them into random subsets appropriate for five-fold-cross-validation. Since the computational cost of the BRkNN classifier depends on the size of the training set used, the CPU time needed for the classification is not reported. The

[4] `http://mulan.sourceforge.net/datasets-mlc.html`

Hamming Loss (HL) is the fraction of the wrong predicted labels to the total number of labels. It is computed as follows:

$$HL = \frac{1}{m} \sum_{i=1}^{m} \frac{|Y_i \Delta Z_i|}{|L|}$$

where m is the number of instances in the dataset, $|L|$ is the number of labels, Y_i is the instances' set of real labels and Z_i is the instances' set of predicted labels. Δ is the symmetric difference of two sets and corresponds to the XOR operation. For example, if the labels of an instance are $\{1, 1, 0, 0, 1\}$ and the predicted labels are $\{1, 1, 0, 1, 0\}$, the Hamming loss will be $\frac{2}{5} = 0.4$. Also, we ran experiments using three different k values (1, 5 and 9).

5.3 Experimental results

Table 2 presents the results obtained by the experimental study. For each dataset, we report Hamming loss and the reduction rate achieved by MRHC. We observe that MRHC achieved reduction rates between 40% and 85%. The average reduction rate is 57.63%. This means that the BRkNN classifier that runs over the condensing set constructed by MRHC is 57.63% faster on average than the BRkNN classifier that runs over the original training set. One could claim that the reduction rates are not very high - certainly, they are lower than the reduction rates achieved by the single-label RHC. We claim that the comparison between the reduction rates achieved on single-label and multi-label datasets does not make sense. Multi-label data is more complex than single-label data. As a result, MRHC cannot identify large homogeneous clusters in multi-label data like RHC does in single-label data.

Moreover, we observe there is no difference in Hamming loss between BRkNN that uses the multi-label condensing set constructed by MRHC and the BRkNN classifier that uses the original training set. BRkNN achieves similar Hamming loss measurements regardless of whether the condensing set or the original training set is used. Therefore, we can safely conclude that MRHC achieves significant gains in reduction rates while accuracy is not negatively affected.

6 Conclusions and future work

Prototype Selection and Generation is an essential pre-processing stage in order to avoid the drawbacks of high computational cost and storage requirements in instance based classification. However, the vast majority of existing PS and PG algorithms are not applicable to multi-label classification problems, and also they can not be effectively used in conjunction with a problem transformation method like Binary Relevance or Label Powerset.

This paper initially presented the recent research efforts for speeding-up the k-NN classifier in the context of multi-label classification. Then, it proposed the MRHC algorithm, which is the first PG algorithm for multi-label training data.

Table 2. Comparison in terms of Hamming Loss (HL (%)) and Reduction Rate (RR (%))

Dataset		**BRkNN** $k=1$	**BRkNN** $k=5$	**BRkNN** $k=9$	**MRHC BRkNN** $k=1$	**MRHC BRkNN** $k=5$	**MRHC BRkNN** $k=9$
CAL	HL:	0.19	0.15	0.14	0.17	0.14	0.14
	RR:	-	-	-	40.50	40.50	40.50
EMT	HL:	0.24	0.20	0.19	0.22	0.20	0.20
	RR:	-	-	-	65.73	65.73	65.73
WQ	HL:	0.38	0.35	0.33	0.37	0.34	0.32
	RR:	-	-	-	40.64	40.64	40.64
SC	HL:	0.13	0.11	0.12	0.12	0.12	0.12
	RR:	-	-	-	85.13	85.13	85.13
YS	HL:	0.24	0.20	0.19	0.23	0.21	0.21
	RR:	-	-	-	51.85	51.85	51.85
BRD	HL:	0.09	0.08	0.08	0.09	0.08	0.09
	RR:	-	-	-	42.70	42.70	42.70
CHD	HL:	0.36	0.33	0.31	0.35	0.32	0.30
	RR:	-	-	-	65.47	65.47	65.47
IMG	HL:	0.29	0.27	0.27	0.28	0.25	0.25
	RR:	-	-	-	71.71	71.71	71.71
MDM	HL:	0.041	0.033	0.032	0.038	0.032	0.032
	RR:	-	-	-	55.86	55.86	55.86

In effect, MRHC is an adaptation of the fast, single-label RHC algorithm. Like RHC, MRHC is parameter-free and is based on a recursive k-means clustering procedure that discovers homogeneous clusters. In the context of multi-label classification, we considered a cluster to be homogeneous when it contains instances with at least one common label. The centroid of each homogeneous cluster constitutes a prototype labeled by the common labels along with each label that appears in the majority of the cluster instances. Thus, MRHC builds a multi-label condensing set that BRkNN can use to search for nearest neighbours and make multi-label predictions. The experimental study used nine multi-label datasets and demonstrated that there is no difference on the accuracy achieved by BRkNN when using the condensing set built by MRHC or the original training set. On the other hand, the CPU time needed for the classification process when using the condensing set is much lower.

This paper showed that Prototype Selection and Generation for multi-label problems is an open research field in the data mining and machine learning context. We plan to adapt well-known single-label PG algorithms on multi-label problems. Next, we plan to developed new parameter-free PS and PG algorithms as well as scalable classification methods for multi-label training sets.

References

1. lvar Arnaiz-Gonzlez, Dez-Pastor, J.F., Rodrguez, J.J., Garca-Osorio, C.: Local sets for multi-label instance selection. Applied Soft Computing **68**, 651–666 (2018). https://doi.org/https://doi.org/10.1016/j.asoc.2018.04.016
2. lvar Arnaiz-Gonzlez, Dez-Pastor, J.F., Rodrguez, J.J., Garca-Osorio, C.: Study of data transformation techniques for adapting single-label prototype selection algorithms to multi-label learning. Expert Systems with Applications **109**, 114–130 (2018). https://doi.org/https://doi.org/10.1016/j.eswa.2018.05.017
3. Brighton, H., Mellish, C.: Advances in Instance Selection for Instance-Based Learning Algorithms. Data Min. Knowl. Discov. **6**(2), 153–172 (Apr 2002). https://doi.org/10.1023/A:1014043630878
4. Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F.: MLeNN: a First Approach to Heuristic Multilabel Undersampling. In: Corchado, E., Lozano, J.A., Quintián, H., Yin, H. (eds.) Intelligent Data Engineering and Automated Learning – IDEAL 2014. pp. 1–9. Springer International Publishing, Cham (2014)
5. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13**(1), 21–27 (1967). https://doi.org/10.1109/TIT.1967.1053964
6. Eleftherios Spyromitros, Grigorios Tsoumakas, I.V.: An Empirical Study of Lazy Multilabel Classification Algorithms. In: Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008) (2008)
7. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (Mar 2012). https://doi.org/10.1109/TPAMI.2011.142
8. Kanj, S., Abdallah, F., Denux, T., Tout, K.: Editing training data for multi-label classification with the k-nearest neighbor rule. Pattern Analysis and Applications **19**(1), 145161 (Feb 2015). https://doi.org/10.1007/s10044-015-0452-8
9. Leyva, E., Gonzlez, A., Prez, R.: Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. Pattern Recognition **48**(4), 1523–1537 (2015). https://doi.org/https://doi.org/10.1016/j.patcog.2014.10.001
10. Ougiaroglou, S., Evangelidis, G.: RHC: non-parametric cluster-based data reduction for efficient k-NN classification. Pattern Analysis and Applications **19**(1), 93–109 (2014). https://doi.org/10.1007/s10044-014-0393-7
11. Skryjomski, P., Krawczyk, B., Cano, A.: Speeding up k-Nearest Neighbors classifier for large-scale multi-label learning on gpus. Neurocomputing **354**, 10–19 (2019). https://doi.org/https://doi.org/10.1016/j.neucom.2018.06.095, recent Advancements in Hybrid Artificial Intelligence Systems
12. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. Trans. Sys. Man Cyber Part C **42**(1), 86–100 (Jan 2012). https://doi.org/10.1109/TSMCC.2010.2103939
13. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. Int J Data Warehousing and Mining **2007**, 1–13 (2007)
14. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data, pp. 667–685. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-0-387-09823-4_34
15. Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Trans. on Systems, Man, and Cybernetics **2**(3), 408–421 (July 1972)

16. Zhang, M.L., Zhou, Z.H.: ML-KNN: a lazy learning approach to multi-label learning. Pattern Recognition **40**(7), 2038–2048 (2007). https://doi.org/https://doi.org/10.1016/j.patcog.2006.12.019